Spoken Language Understanding (SLU) Generative and Discriminant Models comparison

Rodrigo Joni Sestari
number: 179020
rodrigo.sestari@studenti.unitn.it

Abstract

This document has the objective of illustrate the comparison between Generative model, compiled in the previous document and Discriminant models. This document consists in the second project for the course of Language Understanding System of the University of Trento. This project is available on **github.com/rodrigosestari/LUS.git**

1 Introduction

This document describes the steps necessary to implement a Discriminant model using recurrent neural networks and it comparison with Generative model where was used Finite State Transducers. The first part consists in a theory part about Neural Networks, the second one consist to apply the Discriminant model in different parameters, the last part consist to evaluate these models.

2 Dataset structure

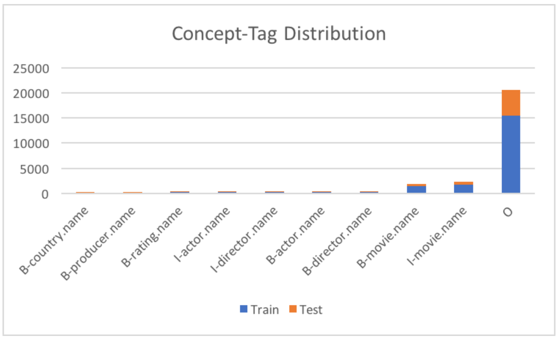
The dataset available to this project are divided by Training and Test:

- **NLSPARQL.train.data**: Dataset used to training the models. The file is format by the tuple <*word, concept*> the concept use **IOB Notation**. one for each line, and each of these sentences is separated by an empty line

- **NLSPARQL.test.data**: Dataset used to test, this dataset has the same structure of NLSPARQL.train.data

- **IOB Notation**: *Inside, Outside, Beginning* is a common tagging format for tagging tokens in a chunking task in computational linguistics. The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk.

Is possible see in the figure bellow the IOB distribution in the dataset:



3 Theory about natural language process

I this chapter introduces some principles about the natural language processing that come used in this document:

- **FST**: *Finite State Transducer,* is a FSA whose state transitions are labelled with both input and output symbols.
- **WFST**: *Weight Finite State Transducer,* A weighted transducer puts weights on transitions in addition to the input and output symbols.
- **LM**: *Language Model* compute the probability of a sentence or sequence of words:

$$P(w_1, \ldots, w_m,) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1})$$

- **Smoothing**: Assume that no n-gram of known words has 0 probability and redistribute probability mass from seen to unseen events, this is known as smoothing.
- **ANN**: *Artificial Neural Network*, is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems.
- **RNN**: *Recurrent Neural Network,* is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs.

4 Tools

To execute the tasks about this document was necessary the following tools:

- **Conlleval**: A Perl script that allow evaluate the SLU result.
- **Theano:** A Python library for efficiently handling mathematical expressions involving multi-dimensional arrays (also known as tensors). It is a common choice for implementing neural network models. Theano has been developed in University of Montreal.

5 RNN Architecture

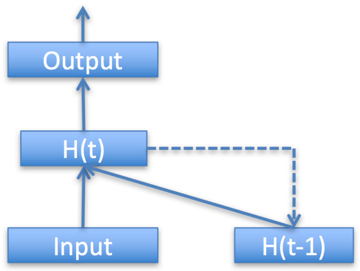
Research on language modelling for speech recognition has increasingly focused on the application of neural networks. Two competing concepts have been developed: On the one hand, feedforward neural networks representing an n-gram approach, on the other hand recurrent neural networks that may learn context dependencies spanning more than a fixed number of predecessor words.

When a feedforward neural network (FFNN) is used, only the direct $(n-1)$ predecessor words w_{i-n+1}^{i-1} are used to predict the probability of the current word w_i. Although it is possible to include words from the previous sentence, most of the time the history is truncated at the beginning of the sentence in the n-gram approach. When a recurrent neural network is used, the full sequence of predecessor words w_1^{i-1} is considered for predicting w_i.

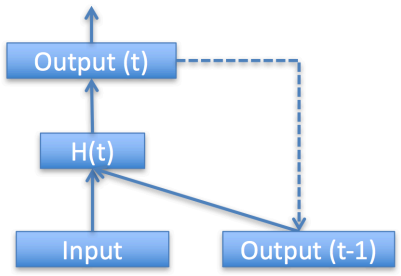
In another way, in a ANN the inputs and outputs are independent between their, the problem is that is not possible to explorer the sequential information. To explorer it, exist the RNN in which, the weighted connections feeding a neuron, also come either from the hidden units of the previous iteration, or from the outputs, been possible consider sequences.

In order to apply a RNN into SLU is used two type of networks:

- **Elman mode**: This network has a connection that feeds the activation of the hidden layer in the previous time step with the current input. The probability of label is estimates by $P(C|W_t, S_{t-1})$ where C is the label, W_t the word an S_{t-1} the previous state.



- **Jordan model**: This network connection that feeds the activation of the output layer at previous time step with the current input. The probability of label is estimates by $P(C|W_t, O_{t-1})$ where C is the label, W_t the word an O the previous output.



6 Implementation

The implementation consists to execute some scripts to training and test the LM, these scripts use an external tool Theano.

6.1 Scripts

Some scripts are implemented to elaborate chain of input/output files. The scripts are:

- **start.sh**: This script call Theano tool in order to execute all models, its requires one parameter, if negative number, run all models, if positive between 1..7, run a specific configuration file.

- **files.sh:** Create the label and word dictionary, the initial training dataset, shuffled and splitted into a training dataset with the 75% of the sentences and in a validation dataset with the remaining 25%.

6.2 Configuration

The configuration file contains these parameters:

- **lr**: starting learning rate
- **win**: context window size
- **bs**: mini batch size
- **nhidden**: size of the hidden layer
- **seed**: random seed
- **emb_dimension**: dimension of the word embedding
- **nepochs**: maximum number of backpropagation steps

Was created 8 configuration files in order to test all parameters, bellow is possible see these models:

mod	lr	win	bs	hid	seed	dim	epoc
1	0.1	9	5	100	3842845	100	25
2	0.1	9	5	100	3842845	100	**10**
3	**0.5**	9	5	100	3842845	100	25
4	0.1	9	5	**200**	3842845	100	25
5	0.1	**7**	5	100	3842845	100	25
6	0.1	9	**10**	100	3842845	100	25
7	0.1	9	5	100	**300**	100	25
8	0.1	9	5	100	3842845	**200**	25

7 Results

The idea is measure the output given by the SLU model.

7.1 Evaluation Methods

- **accuracy**: Accuracy refers to the closeness of a measured value to a standard or known value.

$$Accuracy = \frac{Num.of\ Correct\ Decisions}{Total\ Num.of\ Instances}$$

- **precision**: Precision refers to the closeness of two or more measurements to each other or also called positive predictive value, is the fraction of retrieved instances that are relevant.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- **recall**: recall is the fraction of relevant instances that are retrieved in another way is a measure of how many truly relevant results are returned.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- **FB1**: is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score

$$F_1 = \frac{2 * \ Precision\ * Recall}{(Precisioin\ +\ Recall)}$$

7.2 Evaluation

In the tables bellow is possible see the result of evaluation, the 3 tables are; FST result, got from the first project, Elman and Jordan Networks results, the first column for the FST table represents the smoothing method, instead from the Elman and Jordan represents the model used, the next 3 columns represent the evaluation methods; accuracy, precision, recall and F-measure.

FST

method	accuracy	precision	recall	FB1
absolute	92.88%	76.97%	75.34%	76.15
Katz	92.62%	78.03%	73.88%	75.89
Kneser	92.90%	76.79%	75.53%	76.16
Pre-smo	92.65%	78.41%	74.24%	76.27
Unsmoo	92.54%	78.09%	73.88%	75.93
WittenB	**92.90%**	77.03%	75.62%	76.32

RNN Elman

model	accuracy	precision	recall	FB1
1	**95.00%**	79.20%	73.97%	76.49
2	94.53%	75.53%	71.86%	73.65
3	90.92%	62.01%	55.36%	58.50
4	94.79%	79.04%	73.60%	76.22
5	94.93%	77.99%	74.70%	76.31
6	94.80%	75.76%	72.78%	74.24
7	94.61%	82.22%	76.72%	79.37
8	94.86%	78.71%	72.87%	75.68

RNN Jordan

model	accuracy	precision	recall	FB1
1	94.92%	81.71%	74.52%	77.95
2	94.16%	78.95%	73.24%	75.99
3	88.77%	52.06%	50.87%	51.46
4	94.95%	79.27%	75.34%	77.26
5	94.93%	82.06%	75.07%	78.41
6	94.86%	80.54%	74.34%	77.31
7	**95.53%**	81.84%	77.27%	79.49
8	94.53%	80.44%	73.51%	76.82

8 Conclusion

About RNN results, the Elman and Jordan networks are quite similar between them, but Elman are slightly better than Jordan, because this model takes as input the current input $time_t$ and the previous hidden state $time_{t-1}$.

With a learning rate higher, has the worsen result, I believe because the gradient descent is not fast enough to get better learning results.

About the comparison about the discriminant and generative modes, the RNN model is more accurate than FST, because consider the global decision on a sequence that is more accurate than local decision. The file results can be seen on **https://github.com/rodrigosestari/LUS/tree/master/Second/result**

References

John Wiley and Sons. New York, 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*

Maccartney, B. *(2005).* Stanford. from *https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf*

Mohri et al. (1996) *FSM Toolkit Exercises*

Riccardi (2017). UniTN. Retrieved 10 April, 2017, from http://disi.unitn.it/~riccardi/

Sundermeyer M, et al. (2013) Comparison of feedforward and recurrent neural network language models.